An Implementation of Client-based Retrieval System by Use of Neighborhood Information

近傍情報を用いた分散型情報探索システムの実現

曽 超 (福岡工業短期大学 電子情報システム学科)

An Implementation of Client-based Retrieval System by Use of Neighborhood Information

Chao ZENG (Department of Electronics and Information Systems, Fukuoka Junior College of Technology)

Abstract

The World-Wide Web is a distributed network of information that is spreading over internet sites. Two types of information retrieval systems on the internet have been considered. The first, such as Lycos, AltaVista, use automated WWW robot to built an index database for a part of the WWW and provide an interface for searching information. The second are what called as client-based search tools, such as Fish-Search[1)2)], which do on-line searching on the WWW. In our previous paper[7)], we have discussed some search hueristics and strategies for implementing client-based search tools on large hypertext system such as the WWW. In this paper, we present in detail an implementation of a client-based search system on the WWW using Perl5 and libwww-per15.

Key Words: *WWW, client-based search, strategy, hueristic, neighborhood information*

1. はじめに

インターネットが益々普及する現在，インターネット上に公開される情報の量が爆発的に増加している。インターネット自体が一つの大きな分散型データベースとみることができる。情報提供に最も多く利用されるのは，World-Wide Web (WWW) というネットワーク分散型システムである。WWW サーバによって公開される情報は WWW ブラウザと呼ばれるクライアントソフトによって，その所在 (URL アドレス) さえ分かれば簡単に利用することができる。WWW サーバ上のドキュメントは URL アドレスによって唯一的に定められる。ドキュメントには，他のドキュメントを指し示すリンクを含むことができる。WWW 全体が構造的に一つの大きなハイパーテキスト (Hypertext) を構成している。WWW 上での情報アクセスは，HTTP (HyperText Transfer Protocol) プロトコルに従い，ドキュメントをサーバからクライアントに転送することによって行なわれる。

WWW の利用は WWW ブラウザ(Mosaic, Netscape など) を用いて，あるドキュメントをブラウジングし，さらにその中に含まれるリンクをクリックして他のドキュメントをアクセスする，その繰り返しによって行なわれる。しかし，このような一歩ずつ辿っていく方式は，情報探索には適していない。つまり，必要な情報が何処の誰によって提供されているかの情報が WWW には存在しない。必要な情報を見つけるのに莫大の時間と労力をかかってしまう。このような状況下，WWW 上の必要な情報を探索するためのシステムが幾つか研究開発され，利用されている。

多くの WWW 情報探索システムが「予め用意され

平成9年5月26日受付

たインデックスに対する問い合わせ」(querying pre-computed indexes)の方式をとっている。インデックシングエージェント（indexing agent，あるいはロボット）が自動的にインターネット上を歩き回り，それぞれのWWWサーバにどのような情報が提供されているかを調べ，ドキュメント毎にそれに含まれるキーワードによるインデックスを作る。このように集めたインデックスは，最終的に一箇所に集められ，一つの大きなデータベースとなる。そのデータベースへのキーワードの問い合わせによって情報探索が行なわれる。現在殆んどの探索システムが，例えばAlta Vista，Lycosなど，この方式を採用している。このような探索システムはWWWを利用する際，非常に役立つが，同時に幾つかの欠点が指摘されている。

- 集めるべき情報の量が莫大過ぎて，探索サーバに全ての探索が集中するためサーバの負担があまりにも大きい。
- 全てのインターネット上の情報に対してインデックスを作るのが事実上不可能であり，インデックスは，結局一部の「重要」な情報にしか行なわれていない。
- 探索サーバ上のインデックス情報の老朽化。

上述のような「予め用意されたインデックスに対する問い合わせ」型探索システムの他に，クライアントベース型(client-based)探索システムも提案されている[1)2)]。探索は，与えられたスタートポイントと探索キーワードによって，オンライン的にWWWをアクセスしながら，クライアント側のマシン上で行なわれる。

この論文では，幾つかのWWWドキュメント文章構造に適した探索策略を提案する。また，Perl5とLWP（Library for WWW access in Perl5）を用いてそれらを探索策略をインプリメントしたクライアントベース型探索システムの実現を述べる。LWPは，Perlアプリケーションに WWWへの便利なアクセスインターフェースを提供する。J.W. de Vocht氏の実験的な研究[3)]によって，深さ優先探索(DFS)がハイパーテキスト構造に適している結果がある。しかし，WWWのような規模の大きなハイパーテキストシステムには深さ優先探索は探索時間が非常に長い。オンライン的な探索を実現するためには，適当な策略による探索の誘導が必要である。

以下，本稿で使う幾つかの用語を説明する。ドキュメントとは，URLアドレス付きWWW上のHTML言語ファイルのことをいう。関連ドキュメント（relevant document）とは，探索キーを一回以上含むドキュメントのことをいう。ドキュメントの関連強弱度はそのドキュメントに含むキーワードの数によって評価される。ドキュメントには，リンクを含む。リンクは以下の形のものである。

〈a href="URL"〉ホットテキスト〈/a〉

URLはリンク先のドキュメントのURLアドレスであり，ホットテキストはそのタイトルである。

2．システムのレイアウト

WWWサーバ上のドキュメントはリンクによって互いに参照できる仕組みになっている。情報探索を行なう際に，リンクによってできる探索空間が非常に大きなものとなるのは一般的である。探索空間の一部分に限定して探索を行なうことが時間的に考えると必要である。システムは探索の出発点（start point）と探索キーワードの指定をユーザが行なってから探索を始める。

出発点は一つのURLアドレスで，探索空間のルートとなり，関連情報をリンクしている可能性の大きいドキュメントが望ましい。良い出発点を与えるためには，利用者が探索したい情報に関して一定の知識を持つことが要求される。例えば，Donald E. Knuth先生に関する情報の探索を行なう場合は，Knuth先生がスタンフォード大学の先生なので，出発点はスタンフォード大学のホームページ（http://www.stanford.edu/）が選ぶべきである。更に，もし先生がComputer Science学部の先生であることが知っていれば，出発点はスタンフォード大学Computer Science学部のホームページ（http://www-cs.stanford.edu/）を選ぶべきであろう。明かに，後者はより良い出発点である。もし，所属大学などの情報もない場合はhttp://www.***.edu/のようなアメリカの何処かの大学のホームページを出発点として選ぶ。このような出発点を与えた時，探索空間が大きなものとなり，探索したい情報に辿り着く前に，多くの無関連のページを探索することとなる。

探索キーワードはユーザの持つ目的情報に関する知識との見方もできる。ユーザが如何によい探索のヒントをシステムに与えることが重要である。例えば，先ほどのKnuth先生の例では，「academic compute department knuth engineer faculty」のようなキーワード群が探索によいヒントを与えると考えられる。

このようにキーワードを通じてユーザが知っている探索対象に関する知識をシステムに与えることが探索を成功に導く。実際，人間がブラウザを使って，WWWをブラウジングする時には，そのような目的情報に関連する知識を脳に浮かばせながら，なるべく関連の強いページを先に巡って目的情報に近付けて行くようにしている。

3．探索アルゴリズム

ハイパーテキスト構造における探索には，深さ優先探索が適していることがJ.W. de Vocht氏による実験的な研究[3]によって指摘されている。このシステムも基本探索アルゴリズムとして以下のような深さ優先探索を用いる。

1．出発点URLを探索候補リストに入れる。
2．候補リストの先頭からURLアドレスを一つ取りだし，以下の3．から5．まで繰り返す。
3．そのURLの指すドキュメントをダウンロードする。
4．まず，ダウンロードしてきたドキュメントに関連情報が含まれているかどうかをスキャンして調べる。同時に，中にあるリンクを洗いだし，その中から適当なリンクを選び，一定の順番で候補リストの先頭に追加する。
5．候補リストが空ならば，見つけた情報を出力して終了。空ではなければ，候補リストの先頭からリンクを一つ取り出し，3．へ。

アルゴリズムの効率性は如何に無関連なドキュメントのアクセスをせず，関連情報を見つけることにある。次節では，上述のアルゴリズムを効率良くするための幾つかの探索策略を提案する。

4．探索策略

4．1　探索空間の限定

探索はHTMLテキストドキュメントに限って行なう。画像，クリッカブルマップなどのマルチメディアに含まれる情報を考えない。

探索の策略として，まずURLの構造に注目した。例えば，与えられた出発点はスタンフォード大学のホームページとする。となると，探したい情報がスタンフォード大学と関連のある情報であると考えるが，しかし，スタンフォード大学のホームページから他の大学，あるいは研究機関へのリンクが張られている場合は，探索はすぐにも他の場所に誘導されてしまう。このようなことが基本探索アルゴリズムを非常に効率悪くする場合がある。

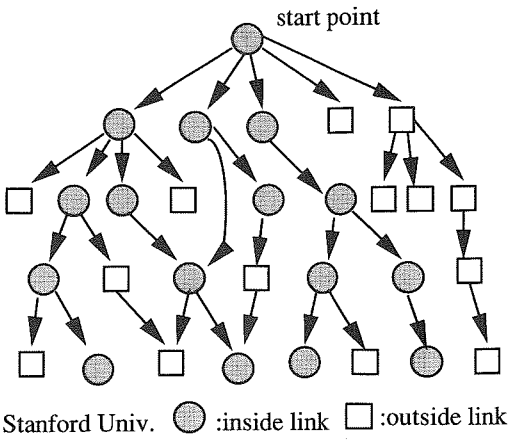

図1　探索策略による探索空間の縮小

システムでは，図1のように，探索空間において，各ドキュメント中のリンクに対して内部，外部リンクに分ける。探索は内部リンクに限って行なわれる。例えば，出発点としてスタンフォード大学のホームページhttp://www.stanford.edu/が与えられた場合，http://*.stanford.edu/pathnameのような同じスタンフォード大学内のリンクだけが探索される。このようなリンクを内部リンクという。また，もしhttp://www.yale.edu/，http://www.ibm.com/などのような外部へのリンクがあるならば，それらを無視する。このようなリンクを外部リンクという。システムでは，内部と外部リンクの区別を出発点のホスト名を用いて，探索開始時にユーザの選択によって決める。

4．2　ホットテキストによる関連評価

ドキュメント内のリンクは以下の形を取っている。

〈a href="URL"〉ホットテキスト〈/a〉

ドキュメントに多くのリンクを含む場合それぞれのホットテキスト部分が次候補を選択する決め手として用いられる。探索キーワードをホットテキストに含むようなリンクを優先的に考えるべきである。リンクの関連強弱度はホットテキストに出現するキーワードの回数によって評価される。

4．3　近傍情報による関連評価

ドキュメントの中には，リンクの前後にそのリンクに関する説明を書く場合が多い。リンクが置かれた位置の近傍（neighborhood）のテキストを利用してそのリンク先と探したい情報との関連性を計ることができる。

ドキュメントにあるリンク（link で記す）の「近傍」（近傍に含む文字数は NH(link) で記す）とは，そのリンクの前後のテキストのことをいう。前後の範囲はドキュメント毎にそれに含む文字の数（P (characters) で記す）とドキュメントに張ってあるリンクの数（P (links) で記す）とによって決める。以下の簡単な式を用いた。

$$\mathrm{NH(link)}=\frac{\mathrm{P}(\mathit{characters})}{\mathrm{P}(\mathit{links})}\times 2$$

式の計算結果が20以下なら，20とする。リンクの近傍にキーワードの出現のチェックによってそのリンクの関連性を評価する。

前節で述べたホットテキストによる関連評価について，もしホットテキストに探索キーワードが含まれていたら，そのリンクの関連性が非常に高いと評価すべきである。しかし，残念ながら実際にはホットテキストに探索キーワードが含まれるケースが少ないため，役立つ時が少ない。近傍情報による関連評価では，関連性こそ多少弱くなるが，そのリンクの前後にあるリンクに関する説明のテキストを評価の対象にするため，実際に探索に役立つ時が多い。

システムでは，ホットテキストによる関連評価と近傍情報による関連評価を結合した探索策略を実現した。リンクのホットテキストと近傍でのキーワードに1回の出現にそれぞれ関連性強弱度 p と q のウェートを付ける。つまり，あるリンクのホットテキストと近傍にキーワードの出現は，それぞれM回とN回とする。そのリンクの探したい情報との関連強弱度は次の式 $p\times \mathrm{M}+q\times \mathrm{N}$ によって評価される。

4．4　部分深さ優先探索

基本探索アルゴリズムは，ドキュメントに含む全てのリンクを候補集合に追加して探索を行なう。展開するリンクの数が多過ぎる場合，メモリオーバーが起こり，探索が途中で止まってしまうことがしばしばある。また，実際に多くのリンクが関連性のないドキュメントを指す場合が多い。こういった無関連ドキュメントを多数展開することで時間的効率も非常に悪くなる。システムには部分深さ優先探索もインプリメントした。部分深さ優先探索では，展開されたドキュメントの関連強弱度により，それから次に考慮すべきリンクの数を決めるようにする。関連性の高いドキュメントに含む多くのリンクが更に探索される価値が高い。その数はドキュメントの展開係数（extending factor）と呼ぶ。

ドキュメントに含むリンクの中から関連性の強い順に展開係数個を選び，候補集合に追加する。リンクの関連性の強弱の評価は上述のホットテキスト関連評価法と近傍情報による関連評価を用いた。

部分深さ優先探索は全ての関連情報を見つけることができないが，探索空間が小さく，よい探索効率を得ることができる。

5．システムの実現

5．1　システム構築ツール

システムの実現は，**Perl5** と **libwww-per5**(LWP) を用いた。日本語テキストページを処理するために，**Kconv** パッケージを利用した。Kconv パッケージには，日本語コード変換のためのメソッドが含まれている。Perl5 と libwww-perl5, 両方ともフリーなものでインターネットを通じて入手し，利用することができる[1]。Perl5 では，モジュール（クラス）の概念が導入され，Perl5 でのプログラミングはオブジェクト指向的に行なうことができるようになった。libwww-perl5 ライブラリは，WWW サーバにアクセスするための API を提供してくれる Perl5 のモジュールの集まりである。**use** Module コマンドがモジュール Module.pm をコンパイル時にロードしてくれる。アロー演算子（→）がオブジェクトのメソッドを呼び出すのに使う。オブジェクトの生成は **new** 演算子を用いて行なう。libwww-perl ライブラリには，主に以下のようなモジュールが含まれる。

LWP::Protocol　各種のプロトコルへのベースインターフェースクラス。http, ftp などの LWP ライブラリがサポートするプロトコルを実現するためのベースクラスとして用いられる。

LWP::UserAgent　WWW ユーザエージェントク

[1] **Perl5**: http://www.perl.com/, **libwww-perl5**: http://www.sn.no/libwww-perl/

ラス。このクラスのオブジェクト（メソッドrequest()）を利用して，WWWサーバに直接リクエスト（HTTP::Requestクラスのインスタンス）を発送（dispatch）することができる。メソッドrequest()の戻り値はクラスHTTP::Responseのインスタンスとなり，そのコンテントはスカラー変数，ファイルまたは他のサブルーチンへ保存することができる。

HTTP::Message　HTTPのメッセージをカプセル化したクラス。HTTPヘッダ，ボディを扱うメソッドが含まれる。クラスHTTP::RequestとHTTP::Responseの親クラスである。

HTTP::Request　HTTPスタイルのリクエストをカプセル化したクラス。このクラスのインスタンスはよくLWP::UserAgentオブジェクトのrequest()メソッドの引数として渡される。

HTTP::Response　HTTPスタイルのレスポンスをカプセル化したクラス。このクラスのインスタンスは普通LWP::UserAgentオブジェクトのrequest()メソッドによって生成される。

HTML::Parse　HTMLドキュメントを解析(parse)するためのモジュールを提供する。ルーチンparse_html()は，引数として与えられたHTMLドキュメントを解析し，LWPの内部表現を用いたHTMLシンタックスツリーを生成する。また，返り値として，HTML::TreeBuilderクラスのインスタンスへのリファレンスを返す。HTML::TreeBuilderは，HTML::Elementのサブクラスである。

HTML::Element　HTMLシンタックスツリーに含まれるリンクを扱うためのメソッドを提供する。

LWPライブラリの詳しい説明については，それぞれのモジュールに含まれるポド（pod）ドキュメント，またはオンライン的にPerl WWW Documentationのページ[2]を参考されたい。

Kconvパッケージ[3]には，日本語文字列に対するコード変換を行なうためのルーチンが提供されている。

kconvルーチン　kconv（文字列，_EUC）のようにkconvルーチンを呼び出し，文字列をEUCコードに変換して，変換した文字列を返り値とする。文字コードの指定には，_EUCの他に_SJISと_JISもある。変換元文字列の文字コードは自動的に判別される。

5．2　システムの実現

システムの基本探索アルゴリズムは，サブルーチンgo_search()の再帰的呼びだしによって実現された。go_searchを呼びだす際，go_search（$target, \%visited）のように，引数として探索ドキュメントのURLと探索したドキュメントを含めたハッシュ変数へのリファレンスを与える。

$targetへのアクセスは，LWP::UserAgentクラス，及びHTTP::Requestクラスのインスタンスを生成し，以下のように行なわれる。

```
my ($ua) = new LWP::UserAgent;
$ua->agent('NETkumo/0.1');
$ua->from('zeng@crown.fjct.fit.ac.jp');
$ua->timeout(120);
my ($req) = new HTTP::Request GET => $target;
my ($res) = $ua->request($req);
```

agent()，from()とtimeout()は，エージェントクラスLWP::UserAgentのメソッドで，それぞれエージェント名，エージェントの所有者とアクセスタイムアウト時間を設定する。request()メソッドは，リクエストを発信するのに用いられる。WWWサーバからのレスポンス$resは，HTTP::Responseのインスタンスである。レスポンスから得たHTMLドキュメント$res → contentに対する以下の二つの処理が行なわれる。

(1) サブルーチンsearch_keys()によるキーワードのチェック：
search_keys（$target, $res → content）のように実行する。日本語キーワードの探索は，EUCコードに変換してから行なうようにしている。キーワードとドキュメントのコード変換は，kconv()によって以下のように行なわれる。

```
for(@key){
    $_ = &kconv($_, _EUC);
}
$doc = &kconv($res → content, _EUC);
```

(2) レスポンスのドキュメントに含まれるリンクの洗いだし：
次の探索リンク候補集は，$res → contentに含

[2]**Perl WWW Documentation**: http://www.perl.com/perl/wwwman/
[3]**Kconv**: ftp://ftp.intec.co.jp/pub/utils/Kconv-1.01.tar.gz

まれるリンクから選ばれる。それらリンクの洗いだしは，以下のように行なわれる。

```
my($p) = parse_html($res→content);
for (@{ $p->extract_links(qw(a)) }) {
     ......
}
```

parse_html()は，モジュールHTML::Parseのルーチンでドキュメント$res→contentを解析する。クラスHTML::Elementのメソッドextract_links()は，指定されたタイプのHTMLリンクを抽出する。また，それらのリンクへのリファレンスを配列にし，その配列へのリファレンスを返してくれる。抽出したいリンクのタイプは，引数で指定する。qw(a)は，<a href="URL">タイプのリンクを指定する。

上記の(2)で得られたリンク集について，外部リンク，既に探索したリンク，破棄（broken）したリンク，及びタイムアウト時間内に返答のないリンクを除いたリンク集を次に探索するリンクの候補集合とする。

探索するリンクの候補集合の中のリンクの探索優先順位を付けるために，ホットテキストによる関連評価と近傍情報による関連評価を用いた。関連評価，及び優先順位付けは，それぞれサブルーチンbefore_after_check()とsort_link()とによって行なわれる。

before_after_check()は，4．3節で述べた近傍情報による関連評価を行なう。ルーチンは，before_after_check($link, $document_content) のように呼び出され，ドキュメント$document_contentの中にある$linkの近傍にキーワードの出現回数を返り値として返す。以下は，before_after_check()サブルーチンである。

```
sub before_after_check {
 my ($ba_fact) = 100;
 my ($ba_count) = 0;
 my ($select_b_text, $select_a_text);
 my ($link_content) = shift @_;
 my ($doc) = &kconv(shift @_, _EUC);
 $doc =~ /$link_content/;
 my ($before_text) = $`;
 my ($after_text) = $';
 my ($center_text) = $&;
 $before_text =~ s/<[^>]*>/ /g;
 $after_text =~ s/<[^>]*>/ /g;
 $before_text =~ s/\s+/ /g;
 $after_text =~ s/\s+/ /g;
 if (length($before_text) < $ba_fact){
   $select_b_text = substr($before_text,
                           -length($before_text));
 }else{
   $select_b_text = substr($before_text, -$ba_fact);
 }
 $select_a_text = substr($after_text, 0, $ba_fact);
 foreach (($select_b_text, $select_a_text)){
  foreach $ckey (@key){
   my (@numckey) =  ($_ =~ /$ckey/gi);
   $ba_count += @numckey;
  }
 }
 return($ba_count);
}
```

sort_link()は，4．2節で述べたホットテキストによる関連評価を行ない，ホットテキストによる関連評価と近傍情報による関連評価の結果を総合してリンクの候補集合について優先順位を付ける。sort_link()は，sort_link(\%link_elem, \%link_ba, \@select links) のように呼び出され，順位付けされた候補リンクの配列@select_linksを生成する。%link_elemと%link_baは，それぞれリンクをキーとするホットテキストと近傍にキーワードの出現回数のハッシュ変数である。以下は，sort_link()サブルーチンである。

```
sub sort_link {
 my ($link_elem_ref) = shift @_;
 my ($link_ba_ref) = shift @_;
 my ($select_links_ref) = shift @_;
 my (%link_key);
 my (%hot_keys);
 foreach $link1 (keys %$link_elem_ref) {
  my ($cnt) = 0;
  foreach $key1 (@key){
   my(@num_key)=(${$link_elem_ref}{$link1}=~/$key1/gi);
   $cnt += @num_key;
  }
  $hot_keys{$link1}=$cnt;
  $link_key{$link1}=$cnt*10+${$link_ba_ref}{$link1};
 }
 my (%link_key1) = %link_key;
 print ``\nall sorted links are:\n'';
 if (keys (%link_key1) == 0) {
  @$select_links_ref = keys %$link_elem_ref;
  print `` No links.\n\n'';
 }else{
  while (keys %link_key1){
   my ($link, @link_key2) = keys %link_key1;
   my ($max) = $link_key1{$link};
   foreach (@link_key2){
    if ($link_key1{$_} > $max) {
     $link = $_;
     $max = $link_key1{$link};
    }
   }
  print `` $link\n[key '';
  print ``appearences in HOTTEXT:$hot_keys{$link}''
    . `` NEIGHBOR: ${$link_ba_ref}{$link}  TOTAL: '' .
    ``$link_key1{$link}(HOTTEXT/NEIGHBOR=10/1) ]\n'';
  push (@$select_links_ref, $link);
  delete $link_key1{$link};
  }
  print ``\n'';
 }
}
```

上記の sort_link()ルーチンは，全てのリンク候補を考慮し，基本探索アルゴリズムを実現している。4．4節で述べた部分深さ優先探索は，サブルーチン select_bread()によって実現される。

部分深さ優先探索は，ドキュメント内のリンク候補集の一部分を選ぶ。ドキュメントの関連性により，関連性の強いドキュメントからはより多いリンクを展開する。ルーチン select_bread()では，展開係数$b_fact を，関連ドキュメントと無関連ドキュメント，それぞれ5と3とする。リンクの順位付けは，ホットテキストによる関連評価と近傍情報による関連評価を総合して行なわれる。ルーチン select_bread()は，select bread(\%link_elem, \%link_ba, \@select_links) のように呼び出され，展開係数$b_fact 個の選ばれた候補リンクの配列@select_links を生成する。以下は，select_bread()ルーチンである。

```
sub select_bread {
 my ($link_elem_ref) = shift @_;
 my ($link_ba_ref) = shift @_;
 my ($select_links_ref) = shift @_;
 my (%link_key);
 my (%hot_keys);
 foreach $link1 (keys %$link_elem_ref) {
  my ($cnt) = 0;
  foreach $key1 (@key){
   my(@num_key)=(${$link_elem_ref}{$link1}=~/$key1/gi);
   $cnt += @num_key;
  }
  $hot_keys{$link1} = $cnt;
  $link_key{$link1}=$cnt*10+${$link_ba_ref}{$link1};
 }
 my (%link_key1) = %link_key;
 print ''\nselected links (at most $b_fact) are:\n'';
 if (keys (%link_key1) == 0) {
  @$select_links_ref = keys %$link_elem_ref;
  print '' No links.\n'';
 }else{
  for($i=1;$i<=$b_fact && keys(%link_key1)!=0;$i++){
   my ($link, @link_key2) = keys %link_key1;
   my ($max) = $link_key1{$link};
   foreach (@link_key2){
    if ($link_key1{$_} > $max) {
     $link = $_ ;
     $max = $link_key1{$link};
    }
   }
   print '' $link \n   [keys '';
   print ''appearences in HOTTEXT: $hot_keys{$link}''
    . ''NEIGHBOR: ${$link_ba_ref}{$link} TOTAL: ''
    . ''$link_key1{$link}(HOTTEXT/NEIGHBOR=10/1)]\n'';
   push (@$select_links_ref, $link);
   delete $link_key1{$link};
  }
 }
}
```

5．3　システムの実行とその評価

システムは現在コマンドラインより利用できる。コマンド NETkumo に第1引数出発点と探索キーワードを付けて実行する。

```
%NETkumo http://www.stanford.edu/ knuth compute academ
        depart engineer faculty
```

実行が開始すると，以下のメッセージが表示され，探索範囲の選択を行なう。

```
番号を選んで探索の範囲を指定してください：
[1. www.stanford.edu  2. stanford.edu  3. edu]
```

1．を選択すると，探索は www.stanford.edu というサーバ上に限定して行なわれる。2．を選択すると，探索は＊.stanford.edu のような WWW サーバ上に限定される。探索の範囲は広くなる。3．を選択すると，探索は更に広範囲に渡って，＊.edu のような WWW サーバ上に限定して行なわれる。範囲の選択の後，探索方法の選択を行なう。

```
探索方法を番号で選んでください：
[1. 部分深さ優先探索  2. 基本深さ優先探索]
```

探索方法の選択が終ると，探索が始まる。中断キーを押すと探索を止めることができる。その時，探索の途中までの結果が出力される。探索結果の一部は以下のようである。

```
                    ⋮
URL: http://soe.stanford.edu/soe.html
HOT TEXT: Stanford University-School of Engineering
TOTAL(keys): 42([knuth: 0][compute: 1][academ: 1]
                [depart: 8][engineer: 32][faculty: 0])

URL: http://www-cs.stanford.edu/
HOT TEXT: Computer Science Department
TOTAL(keys): 18([knuth: 0][compute: 7][academ: 0]
                [depart: 6][engineer: 2][faculty: 3])

URL: http://www-cs.stanford.edu/People/faculty.html
HOT TEXT: Faculty
TOTAL(keys): 13([knuth: 2][compute: 0][academ: 0]
                [depart: 1][engineer: 0][faculty: 10])

                    ⋮

URL: http://www-cs-faculty.Stanford.EDU/~knuth/
HOT TEXT: http://www-cs-faculty.Stanford.EDU/~knuth/
TOTAL(keys): 4([knuth: 1][compute: 3][academ: 0]
               [depart: 0][engineer: 0][faculty: 0])

                    ⋮
```

システムはインターネット上の WWW サーバをアクセスしながら，探索を行なうから，探索の時間的効率は WWW サーバへのアクセス時間に強く依存する

が，ホットテキストによる関連評価と近傍情報による関連評価を探索策略として用いることによって，システムが目的情報と無関連のドキュメントのアクセスを減少することに成功している。また，部分深さ優先探索の実現により，システムのオンライン性をさらにあげる効果が得られた。

6．結　　言

本論文では，WWW上の情報探索システムを構築する際，情報の関連性評価について，ホットテキストによる関連評価と近傍情報による関連評価を提案した。また，それらの評価方法をインプリメントしたクライアントベース型情報探索システムの実現を行ない，ハイパーテキスト構造に適している策略である結果が得られた。これらの策略を用いることにより，システムは関連性の高いドキュメントを優先的に探索し，探索の効率を上げることができた。関連性の評価は単純にドキュメントでのキーワードの出現によって評価されているが，今後の課題として単語，あるいは文章の意味解析による情報関連性評価などが考えられる。

最後に，共同ゼミを通じて貴重な助言を下さった九州工業大学の周　能法助教授に感謝したい。

参 考 文 献

[1] P.M.E. De Bra and R.D.J. Post: “Searching for Arbitrary Information in the WWW: the Fish-Search for Mosaic”, 2nd International World-Wide Web Conference, http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/debra/article.html.

[2] P.M.E. De Bra and R.D.J. Post: “Information Retrieval in the World-Wide Web: Making client-based searching feasible”, Computer Networks and ISDN Systems, 1994, 183-192.

[3] J.W. de Vocht: “Experiments for the characterization of hypertext structures”, Master’s thesis, Eindhoven Univ. of Technology, Apr. 194.

[4] T. Berners-Lee, R. Fielding and H. Frystyk: “Hypertext Transfer Protocol—HTTP/1.0”, RFC1945, May 1996.

[5] E. Berk, and J. Devlin (Eds.): “Hypertext/Hypermedia Handbook”, New York: McGraw-Hill, 1991.

[6] N.F. Zhou and C. Zeng: “On Cooperative Search in Logic Programming”, 信学技報 Vol.95 No.211, 1995, 25-31.

[7] 曽　超：“インターネットにおける分散型情報探索システムの試作”，1997年度人工知能学会全国大会（第11回）論文集，1997，474-477.